

RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	FFFFFFFFFFF	FFFFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	FFFFFFFFFFF	FFFFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	FFFFFFFFFFF	FFFFFFFFFFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	FFF	FFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	FFFFFFFFF	FFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	FFFFFFFFF	FFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	FFFFFFFFF	FFFFFFFFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF	
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF	
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF	

```
PPPPPPPP  AAAAAA  RRRRRRRR  SSSSSSSS  EEEEEEEEE  PPPPPPPP
PPPPPPPP  AAAAAA  RRRRRRRR  SSSSSSSS  EEEEEEEEE  PPPPPPPP
PP      PP  AA      AA  RR      RR  SS      SS  EEEEEEEEE  PP      PP
PP      PP  AA      AA  RR      RR  SS      SS  EEEEEEEEE  PP      PP
PP      PP  AA      AA  RR      RR  SS      SS  EEEEEEEEE  PP      PP
PP      PP  AA      AA  RRRRRRRR  SSSSSS  EEEEEEEEE  PPPPPPPP
PPPPPPPP  AA      AA  RRRRRRRR  SSSSSS  EEEEEEEEE  PPPPPPPP
PP      AA  AAAAAAAAAA  RR  RR  SS      SS  EEEEEEEEE  PP
PP      AA  AAAAAAAAAA  RR  RR  SS      SS  EEEEEEEEE  PP
PP      AA      AA  RR      RR  SS      SS  EEEEEEEEE  PP
PP      AA      AA  RR      RR  SSSSSSSS  SSSSSSSS  EEEEEEEEE  PP
PP      AA      AA  RR      RR  SSSSSSSS  SSSSSSSS  EEEEEEEEE  PP
```

```
LL  I I I I I  SSSSSSSS
LL  I I I I I  SSSSSSSS
LL  I I      SS
LL  I I      SS
LL  I I      SS
LL  I I      SS
LL  I I      SSSSSS
LL  I I      SSSSSS
LL  I I      SS
LL  I I      SS
LL  I I      SS
LL  I I      SS
LL  I I      SSSSSSSS
LLLLLLLLLLL  I I I I I  SSSSSSSS
LLLLLLLLLLL  I I I I I  SSSSSSSS
```



```
1 0001 0 %TITLE 'Convert page number to internal format'
2 0002 0 MODULE PARSEP ( IDENT = 'V04-000'
3 P 0003 0 %BLISS32[
4 0004 0 ADDRESSING_MODE(EXTERNAL=LONG_RELATIVE, NONEXTERNAL=LONG_RELATIVE)
5 0005 0 ]
6 0006 0 ) =
7 0007 1 BEGIN
8 0008 1
9 0009 1 *****
10 0010 1 *
11 0011 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
12 0012 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
13 0013 1 * ALL RIGHTS RESERVED.
14 0014 1 *
15 0015 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
16 0016 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
17 0017 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
18 0018 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
19 0019 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
20 0020 1 * TRANSFERRED.
21 0021 1 *
22 0022 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
23 0023 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
24 0024 1 * CORPORATION.
25 0025 1 *
26 0026 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
27 0027 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
28 0028 1 *
29 0029 1 *
30 0030 1 *****
31 0031 1
32 0032 1 ++
33 0033 1 FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS
34 0034 1
35 0035 1 ABSTRACT: Translates a 'human-readable' page number into internal format.
36 0036 1
37 0037 1
38 0038 1 ENVIRONMENT: Transportable
39 0039 1
40 0040 1 AUTHOR: R.W.Friday CREATION DATE: February 1979
41 0041 1
```

PARSEP
V04-000

Convert page number to internal format
Revision History

M 4
16-Sep-1984 01:27:09
14-Sep-1984 13:07:44

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]PARSEP.BLI;1

Page 2
(2)

:	43	0042	1	%SBTTL 'Revision History'	
:	44	0043	1		
:	45	0044	1	MODIFIED BY:	
:	46	0045	1		
:	47	0046	1	005	REM00005 Ray Marshall 21-Mar-1984
:	48	0047	1		Implemented the foreign language conditionals for compiling
:	49	0048	1		fixed output words. At this time, we only have the German
:	50	0049	1		translations available. But, since the German word for INDEX
:	51	0050	1		is the same as in English, that conditional isn't used.
:	52	0051	1		
:	53	0052	1	004	KAD00004 Keith Dawson 07-Mar-1983
:	54	0053	1		Global edit of all modules. Updated module names, idents,
:	55	0054	1		copyright dates. Changed require files to BLISS library.
:	56	0055	1		
:	57	0056	1	--	


```

: 59      0057 1 %SBTTL 'Module Level Declarations'
: 60      0058 1
: 61      0059 1 TABLE OF CONTENTS:
: 62      0060 1
: 63      0061 1
: 64      0062 1 INCLUDE FILES:
: 65      0063 1
: 66      0064 1
: 67      0065 1 LIBRARY 'NXPORT:XPORT';      ! XPORT Library
: 68      0066 1 REQUIRE 'REQ:RNODEF';      ! RUNOFF variant definitions
: 69      0197 1
: 70      U 0198 1 %IF DSRPLUS %THEN
: 71      U 0199 1 LIBRARY 'REQ:DPLLIB';      ! DSRPLUS BLISS Library
: 72      0200 1 %ELSE
: 73      0201 1 LIBRARY 'REQ:DSRLIB';      ! DSR BLISS Library
: 74      0202 1 %FI
: 75      0203 1
: 76      0204 1
: 77      0205 1 MACROS:
: 78      0206 1
: 79      0207 1
: 80      0208 1 EQUATED SYMBOLS:
: 81      0209 1
: 82      0210 1
: 83      0211 1 OWN STORAGE:
: 84      0212 1
: 85      0213 1
: 86      0214 1 EXTERNAL REFERENCES:
: 87      0215 1
: 88      0216 1 EXTERNAL LITERAL      !Error messages
: 89      0217 1 RNFCRP;
: 90      0218 1
: 91      0219 1 EXTERNAL
: 92      0220 1 FS01 : FIXED_STRING,
: 93      0221 1 KHAR;
: 94      0222 1
: 95      0223 1 EXTERNAL ROUTINE
: 96      0224 1 CONVLB,
: 97      0225 1 ERM,
: 98      0226 1 GETNUM,
: 99      0227 1 GSLU;
: 100     0228 1

```



```

: 102      0229 1 GLOBAL ROUTINE PARSEP (IRA, PAGEN) =
: 103      0230 1
: 104      0231 1 !++
: 105      0232 1 FUNCTIONAL DESCRIPTION:
: 106      0233 1
: 107      0234 1     PARSEP scans IRA in an attempt to recognize a page number.
: 108      0235 1     If a page number is recognized, the results are returned
: 109      0236 1     in PAGEN, and PARSEP returns TRUE as its completion code.
: 110      0237 1     If a page number is not recognized, PAGEN is left in an
: 111      0238 1     undefined state, and PARSEP returns false.
: 112      0239 1
: 113      0240 1 FORMAL PARAMETERS:
: 114      0241 1
: 115      0242 1     IRA is the FIXED_STRING to be scanned. PAGEN is where
: 116      0243 1     the results go.
: 117      0244 1
: 118      0245 1 IMPLICIT INPUTS:
: 119      0246 1
: 120      0247 1     The format of the page number is implicit to this routine.
: 121      0248 1     The required format is:
: 122      0249 1     SECTION-PAGE SUBPAGE
: 123      0250 1     where SECTION is a digit or string of letters, thereby indicating
: 124      0251 1     a chapter, appendix, or the index,
: 125      0252 1     where the '-' is required iff SECTION is present,
: 126      0253 1     where PAGE is required if SECTION is a number, and indicates
: 127      0254 1     the page number,
: 128      0255 1     where SUBPAGE, if present, is a single letter.
: 129      0256 1
: 130      0257 1 IMPLICIT OUTPUTS:      None
: 131      0258 1
: 132      0259 1 ROUTINE VALUE:
: 133      0260 1 COMPLETION CODES:
: 134      0261 1
: 135      0262 1     Returns TRUE if a page number was recognized, and it was ok.
: 136      0263 1     Returns FALSE if no page number was found, or it was bad.
: 137      0264 1
: 138      0265 1 SIDE EFFECTS: None
: 139      0266 1
: 140      0267 1 --
: 141      0268 1
: 142      0269 2 BEGIN
: 143      0270 2
: 144      0271 2 MAP
: 145      0272 2     IRA : REF FIXED_STRING,
: 146      0273 2     PAGEN : REF PAGE_DEFINITION;
: 147      0274 2
: 148      0275 2 LOCAL
: 149      0276 2     HOLD_NUMBER,
: 150      0277 2     NUMBER_VALUE,
: 151      0278 2     NUMBER_SIGN,
: 152      0279 2     NUMBER_LENGTH;
: 153      0280 2
: 154      0281 2
: 155      0282 2 !Try for a simple page number.
: 156      0283 2 IF NOT GETNUM (.IRA, NUMBER_VALUE, NUMBER_SIGN, NUMBER_LENGTH)
: 157      0284 2 THEN
: 158      0285 2 !Bad number (as opposed to none). Give up.

```



```

159 0286 3 BEGIN
160 0287 3 ERM (RNFCRP, 0, 0);
161 0288 3 RETURN FALSE;
162 0289 3 END
163 0290 3 ELSE
164 0291 3 IF .NUMBER_LENGTH GTR 0
165 0292 3 THEN
166 0293 3 !A number was specified.
167 0294 3 !Don't know if it's a page number or chapter number yet.
168 0295 3 BEGIN
169 0296 3
170 0297 3 IF .NUMBER_VALUE LSS 0
171 0298 3 THEN
172 0299 3 !A negative number is illegal. Give up.
173 0300 3 BEGIN
174 0301 3 ERM (RNFCRP, 0, 0);
175 0302 3 RETURN FALSE;
176 0303 3 END;
177 0304 3
178 0305 3 HOLD_NUMBER = .NUMBER_VALUE;
179 0306 3 !If another number follows, the first number was a
180 0307 3 !a chapter number. Otherwise it was a page number.
181 0308 3
182 0309 3 !The following code catches the case of "l-i" ... which would cause ERMA to
183 0310 3 !wander out into space, if GETNUM is allowed to call ERMA (RNFMFN ...).
184 0311 3 IF .KHAR EQL 'C'
185 0312 3 AND NOT DIGIT (CH$RCHAR (.FS_NEXT (IRA)))
186 0313 3 THEN
187 0314 3 BEGIN
188 0315 3 ERM (RNFCRP, 0, 0);
189 0316 3 RETURN FALSE;
190 0317 3 END;
191 0318 3
192 0319 3 IF NOT GETNUM (.IRA, NUMBER_VALUE, NUMBER_SIGN, NUMBER_LENGTH)
193 0320 3 THEN
194 0321 3 !Bad number. Give up.
195 0322 3 BEGIN
196 0323 3 ERM (RNFCRP, 0, 0);
197 0324 3 RETURN FALSE;
198 0325 3 END
199 0326 3 ELSE
200 0327 3 IF .NUMBER_LENGTH GTR 0
201 0328 3 THEN
202 0329 3 !Got a page number. Save it.
203 0330 3 BEGIN
204 0331 3 PAGEN [SCT_TYP] = SCT_CHAPT;
205 0332 3 PAGEN [SCT_NUMBER] = .HOLD_NUMBER;
206 0333 3 PAGEN [SCT_PAGE] = ABS (.NUMBER_VALUE);
207 0334 3 !Flow now merges with looking for a subpage.
208 0335 3 END
209 0336 3 ELSE
210 0337 3 !There is no second number, so first number
211 0338 3 !is really a page number.
212 0339 3 BEGIN
213 0340 3 PAGEN [SCT_PAGE] = .HOLD_NUMBER
214 0341 3 !Flow now merges with looking for a subpage.
215 0342 3 END

```



```

216      0343 3      END
217      0344 2      ELSE
218      0345 2      !No leading number, so try for letters.
219      0346 3      BEGIN
220      0347 3      FS_INIT (FS01);
221      0348 3
222      0349 3      IF GSLU (.IRA, FS01) NEQ GSLU_NORMAL
223      0350 3      THEN
224      0351 3      !User said something strange. Give up.
225      0352 4      BEGIN
226      0353 4      ERM (RNFCRP, 0, 0);
227      0354 4      RETURN FALSE;
228      0355 4      END
229      0356 3      ELSE
230      0357 3      !Got something starting with a letter.
231      0358 4      BEGIN
232      0359 4      LOCAL
233      0360 4      index_string_length :
234      U 0361 4 %IF french %THEN
235      U 0362 4 INITIAL(5);
236      0363 4 %ELSE
237      U 0364 4 %IF italian %THEN
238      U 0365 4 INITIAL(5);
239      0366 4 %ELSE ! The German and English words are the same length.
240      0367 4 INITIAL(5);
241      0368 4 %FI %FI
242      0369 4
243      0370 4 IF .FS_LENGTH (FS01) EQL .index_string_length
244      0371 4 THEN
245      0372 4 IF CH$EQL ( .index_string_length
246      0373 4 .,FS_START (FS01)
247      0374 4 .,index_string_length
248      U 0375 4 %IF french %THEN
249      U 0376 4 .,CH$PTR (UPLIT ('INDEX')))
250      0377 4 %ELSE
251      U 0378 4 %IF italian %THEN
252      U 0379 4 .,CH$PTR (UPLIT ('INDEX')))
253      0380 4 %ELSE ! German and English are the same word here:
254      0381 4 .,CH$PTR (UPLIT ('INDEX')))
255      0382 4 %FI %FI
256      0383 4 THEN
257      0384 4 !User said /ORANGE:"INDEX...."
258      0385 4 !Merge with logic to get a page number.
259      0386 4 PAGEN [SCT_TYP] = SCT_INDEX
260      0387 4 ELSE
261      0388 4 !There are no 5 character appendix names. Give up.
262      0389 5 BEGIN
263      0390 5 ERM (RNFCRP, 0, 0);
264      0391 5 RETURN FALSE;
265      0392 5 END
266      0393 4 ELSE
267      0394 4 !Found an appendix letter
268      0395 5 BEGIN
269      0396 5
270      0397 5 !Convert string to equivalent binary value and save it.
271      0398 5 PAGEN [SCT_NUMBER] = CONVLB(.FS_START(FS01), .FS_LENGTH(FS01));
272      0399 5 PAGEN [SCT_TYP] = SCT_APPEND;

```



```

: 273 0400 5
: 274 0401 4
: 275 0402 4
: 276 0403 4
: 277 0404 4
: 278 0405 4
: 279 0406 4
280 0407 4
281 0408 5
282 0409 5
283 0410 5
284 0411 5
285 0412 4
286 0413 4
287 0414 4
288 0415 4
289 0416 5
290 0417 5
291 0418 5
292 0419 5
293 0420 4
294 0421 4
295 0422 3
296 0423 2
297 0424 2
298 0425 2
299 0426 2
300 0427 2
301 0428 2
302 0429 2
303 0430 2
304 0431 2
305 0432 2
306 0433 2
: 307 0434 1

      END;
      !Attempt to get a page number
      IF NOT GETNUM (.IRA, NUMBER_VALUE, NUMBER_SIGN, NUMBER_LENGTH)
      THEN
      !Bad number. Give up.
      BEGIN
      ERM (RNFCRP, 0, 0);
      RETURN FALSE;
      END
      ELSE
      IF .NUMBER_LENGTH GTR 0
      THEN
      BEGIN
      PAGEN [SCT_PAGE] = ABS (.NUMBER_VALUE);
      !Merge with logic to get a subpage.
      END
      ELSE
      PAGEN [SCT_PAGE] = 1;
      END;
      END;

      !Try to get a subpage.
      FS INIT (FS01);
      GSLU (.IRA, FS01);

      IF .FS_LENGTH (FS01) NEQ 0
      THEN
      PAGEN [SCT_SUB_PAGE] = CONVLB (.FS_START(FS01), .FS_LENGTH(FS01));

      RETURN TRUE;
      END;

```

!Success.
!End of PARSEP

```

.TITLE PARSEP Convert page number to internal format
.IDENT \V04-000\

.PSECT $PLIT$,NOWRT,NOEXE,2

00 00 00 58 45 44 4E 49 00000 P.AAA: .ASCII \INDEX\<0><0><0> ;

.EXTRN RNFCRP, FS01, KHAR
.EXTRN CONVLB, ERM, GETNUM
.EXTRN GSLU

.PSECT $CODE$,NOWRT,2

.ENTRY PARSEP, Save R2,R3,R4,R5,R6,R7,R8 : 0229
MOVAB CONVLB, R8
MOVAB GSLU, R7
MOVAB GETNUM, R6
MOVAB FS01, R5
SUBL2 #12, SP
PUSHL SP : 0283

```

```

01FC 00000
58 00000000G EF 9E 00002
57 00000000G EF 9E 00009
56 00000000G EF 9E 00010
55 00000000G EF 9E 00017
5E 0C C2 0001E
5E DD 00021

```

			08	AE	9F	00023	PUSHAB	NUMBER_SIGN	
			10	AE	9F	00026	PUSHAB	NUMBER_VALUE	
		54	04	AC	D0	00029	MOVL	IRA, R4	
					54	DD	PUSHL	R4	
		66		04	FB	0002F	CALLS	#4, GETNUM	
		09		50	E9	00032	BLBC	R0, 1\$	
				6E	D5	00035	TSTL	NUMBER_LENGTH	0291
				50	15	00037	BLEQ	5\$	
			08	AE	D5	00039	TSTL	NUMBER_VALUE	0297
				03	18	0003C	BGEQ	2\$	
				009E	31	0003E	BRW	8\$	
		52	08	AE	D0	00041	MOVL	NUMBER_VALUE, HOLD_NUMBER	0305
		2D	00000000G	EF	D1	00045	CMPL	KHAR, #45	0311
				0C	12	0004C	BNEQ	3\$	
		30	04	B4	91	0004E	CMPB	@4(R4), #48	0312
				EA	1F	00052	BLSSU	1\$	
		39	04	B4	91	00054	CMPB	@4(R4), #57	
				E4	1A	00058	BGTRU	1\$	
				5E	DD	0005A	PUSHL	SP	0319
			08	AE	9F	0005C	PUSHAB	NUMBER_SIGN	
			10	AE	9F	0005F	PUSHAB	NUMBER_VALUE	
				54	DD	00062	PUSHL	R4	
		66		04	FB	00064	CALLS	#4, GETNUM	
		75		50	E9	00067	BLBC	R0, 8\$	
		50	08	AC	D0	0006A	MOVL	PAGEN, R0	0331
				6E	D5	0006E	TSTL	NUMBER_LENGTH	0327
				11	15	00070	BLEQ	4\$	
50				01	F0	00072	INSV	#1, #0, #4, (R0)	0331
	04	00		52	D0	00077	MOVL	HOLD_NUMBER, 4(R0)	0332
		A0							
		51	08	AE	D0	0007B	MOVL	NUMBER_VALUE, R1	0333
				7D	19	0007F	BLSS	10\$	
				7E	11	00081	BRB	11\$	
		08	A0	52	D0	00083	MOVL	HOLD_NUMBER, 8(R0)	0340
				7C	11	00087	BRB	12\$	0319
									0347
			0C	A5	D4	00089	CLRL	FS01+12	
		65							
		A5	04	A5	9E	0008C	MOVAB	FS01+16, FS01	
				65	D0	00090	MOVL	FS01, FS01+4	
				30	BB	00094	PUSHR	#^M<R4,R5>	0349
		67		02	FB	00096	CALLS	#2, GSLU	
		01		50	D1	00099	CMPL	R0, #1	
				41	12	0009C	BNEQ	8\$	
		50		05	D0	0009E	MOVL	#5, INDEX_STRING_LENGTH	0358
		50	0C	A5	D1	000A1	CMPL	FS01+12, INDEX_STRING_LENGTH	0370
				13	12	000A5	BNEQ	6\$	
		00000000'	EF	00	B5	50	CMPC3	INDEX_STRING_LENGTH, @FS01, P.AAA	0372
						2D	BNEQ	8\$	
08	BC		04	00	02	F0	INSV	#2, #0, #4, @PAGEN	0386
					15	11	BRB	7\$	
		52		08	AC	D0	MOVL	PAGEN, R2	0398
				0C	A5	DD	PUSHL	FS01+12	
					65	DD	PUSHL	FS01	
		68		02	FB	000C3	CALLS	#2, CONVLB	
		A2		50	D0	000C6	MOVL	R0, 4(R2)	
62			04	03	F0	000CA	INSV	#3, #0, #4, (R2)	0399
		00		5E	DD	000CF	PUSHL	SP	0405
			08	AE	9F	000D1	PUSHAB	NUMBER_SIGN	
			10	AE	9F	000D4	PUSHAB	NUMBER_VALUE	

			54	DD	000D7	PUSHL	R4		
			04	FB	000D9	CALLS	#4, GETNUM		
			50	E8	000DC	BLBS	R0, 9\$		
			7E	7C	000DF	CLRQ	-(SP)		0409
		00000000G	8F	DD	000E1	PUSHL	#RNF CRP		
		EF	03	FB	000E7	CALLS	#3, ERM		
			42	11	000EE	BRB	16\$		0410
			50	08	AC	DC	000F0	9\$:	0417
			6E	D5	000F4	TSTL	PAGEN, R0		0414
			0F	15	000F6	BLEQ	NUMBER_LENGTH		
			51	08	AE	D0	000F8	MOVL	NUMBER_VALUE, R1
			03	18	000FC	BGEQ	11\$		0417
			51		51	CE	000FE	10\$:	
08			A0		51	D0	00101	11\$:	
					04	11	00105	12\$:	
08			A0		01	D0	00107	13\$:	0414
					A5	D4	0010B	14\$:	0421
			65	0C	A5	9E	0010E		0426
			A5	10	65	D0	00112		
04					30	BB	00116		0427
			67		02	FB	00118		
			50	0C	A5	D0	0011B		0429
					0D	13	0011F		
					50	DD	00121		0431
					65	DD	00123		
08	BC		68		02	FB	00125		
			10		50	F0	00128		
			50		01	D0	0012E	15\$:	0433
					04	00131			
					50	D4	00132	16\$:	0434
					04	00134			
						RET			

; Routine Size: 309 bytes, Routine Base: \$CODE\$ + 0000

: 308 0435 1
: 309 0436 1 END
: 310 0437 0 ELUDOM

!End of module

PSECT SUMMARY

Name	Bytes	Attributes
\$PLITS	8	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)
\$CODE\$	309	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

----- Symbols ----- Pages Processing

PARSEP
V04-000

Convert page number to internal format
Module Level Declarations

H 5
16-Sep-1984 01:27:09
14-Sep-1984 13:07:44

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]PARSEP.BLI;1

Page 10
(4)

File	Total	Loaded	Percent	Mapped	Time
\$255\$DUA28:[SYSLIB]XPORT.L32;1	590	0	0	252	00:00.1
-\$255\$DUA28:[RUNOFF.SRC]DSRLIB.L32;1	1248	21	1	86	00:00.2

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:PARSEP/OBJ=OBJ\$:PARSEP MSRC\$:PARSEP/UPDATE=(ENH\$:PARSEP)

; Size: 309 code + 8 data bytes
; Run Time: 00:07.7
; Elapsed Time: 00:15.6
; Lines/CPU Min: 3427
; Lexemes/CPU-Min: 14196
; Memory Used: 101 pages
; Compilation Complete

0347

AH-BT13A-SE
 VAX/VMS V4.0

**DIGITAL EQUI
CONFIDENTIAL**

MENT CORPORATION
AND PROPRIETARY